基于增量最短路径优先算法的高效 LFA 实现方法 *

耿海军 [1]，郭小英 [1]，尹 霞 [2]

(1. 山西大学 软件学院，太原 030006; 2. 清华大学 计算机科学与技术，北京 100084)

摘 要：针对已有 LFA 实现方式计算开销大和部署难度高的问题，提出了一种基于增量最短路径优先算法的 LFA 实现方法(LFA implementation method based on incremental shortest path first algorithm，ERPISPF)。首先将快速实现 LFA 的问题转换为如何在以计算节点为根的最短路径树上高效的计算其所有邻居节点到网络其余所有节点的最小代价问题，然后提出了计算该代价的定理并且证明了它的正确性，最后从理论上分析了算法的时间复杂度。仿真结果表明，ERPISPF 不仅计算开销小，并且与 LFC 的故障保护率是相同的。

关键词：增量最短路径优先；LFA 规则；网络故障

中图分类号：TP301.6 doi: 10.19734/j.issn.1001-3695.2018.09.0646

Efficient LFA implementation method based on incremental shortest path first algorithm

Geng Haijun[1], Guo Xiaoying[1], Yin Xia[2]

(1. *School of Software Engineering*, *Shanxi University*, *Taiyuan* 030006, *China*; 2. *Dept. of Computer Science & Technology*, *Tsinghua University*, *Beijing* 100084, *China*)

Abstract: An efficient LFA implementation method which is based on Incremental Shortest Path First Algorithm (ERPISPF) was proposed to reduce the computational overhead and deployment difficulty of the existing LFA algorithm. The paper first turns the problem of quick implementation of LFA into how to efficiently calculate the minimum cost of all its neighbors to all other nodes of the network on the shortest path tree rooted at the compute node. Then a theorem for calculating the cost is presented and its correctness is proved. Finally, the time complexity of the algorithm was theoretically analyzed. Experiments show that compared with LFA algorithm, ERPISPF not only has less computation overhead, but also provides the same failure protection rate as LFA.

Key words: i-SPF; loop free alternates; network failures

随着互联网的飞速发展，新的应用程序对互联网提出了更加苛刻的要求。如何保证数据的无间断传输成为科研工作们研究的重要课题[1~3]。从互联网诞生之日开始，学术界和工业界就开始研究如何提高域内路由协议对故障的恢复能力[4~6]。目前，被公认的方法可以分为两种类型，分别是被动恢复方法和主动保护方法。被动恢复方法也称为事后恢复方案，即当网络中出现故障后，路由协议重新收敛，重新计算路由表。针对该方案的研究主要包括，利用增量最短路径优先算法降低计算最短路径树的时间[7,8]；通过调整协议的默认参数来加速故障的检测速度[9]；限制故障的洪泛范围[10]；在数据包的包头中携带故障信息[11]；修改网络中链路的权值避免路由环路[12]；规定报文更新次序[13]等。然而上述方案容易导致路由不稳定，出现路由震荡等问题。主动保护方法也称为事先预防方案[14]，即提前计算出所有节点对之间的备份路径。当网络出现故障时，利用事先计算出的备份路径转发受故障影响的报文，从而尽可能降低网络的中断时间。在过去的几十年里，大部分的研究主要集中在主动保护方法方面，因此本文也将重点研究这种类型的方案。在主动保护方法中，有一部分研究是基于标签技术的，例如 Not-Via[15]、隧道[16]、MPLS[17]和 Segment Routing[18]等。然而这部分研究对网络协议的改动较大，因此不是本文研究的重点。

LFA[19]是一种具有较强竞争力的路由保护方案,本文研究如何高效的实现 LFA，降低互联网服务提供商部署 LFA 的开销。

0 相关工作

文献[20]中提出首先利用无环路条件计算节点对之间所有符合条件的下一跳，然后使用标签技术选择合适的下一跳转发报文。但是该方案的算法复杂度较高，需要修改报文的头部，所有该方案不适合在实际网络中使用。FIR[21]通过设置时间参数来抑制故障信息的发送，利用本地备份下一跳转发受影响的报文。如果在设定的时间内，网络中的故障依然没有恢复，则洪泛故障信息，重新进行路由收敛。FIR 适用于临时故障的恢复，对于长时间故障的效果较差，可以保护网络中的链路故障，无法应对节点故障，并且抑制参数的选取是一个难题。文献[22]对传统 FIR 进行了调整，使其可以保护节点故障情形。文献[23]对 FIR 进行了进一步的研究，使其支持链路权值不对称的网络。MRC[24]采用多拓扑的思想配置路由，从而保护网络中的故障。

IETF 提出利用 LFA 方法解决网络中的故障。然而计算节点需要计算多个最短段路径树来实现 LFA，大大加剧了路由器的负担。为了降低实现 LFA 的算法开销，相关人员做了大量的工作，比较典型的研究包括 TBFH[25]和 DMPA[26]。

收稿日期：2018-09-25；**修回日期：**2018-10-25 **基金项目：**国家自然科学基金资助项目（61702315）

作者简介：耿海军（1983-），男，山西灵石人，讲师，博士，主要研究方向为路由算法和网络体系结构等（ghj123025449@163.com）；郭小英（1985-），女，山西原平人，讲师，博士，主要研究方向为机器学习、人工智能等；尹霞（1972-），女，北京人，教授，博士，主要研究方向为网络协议测试和路由算法等.

1　网络模型和问题描述

1.1　网络模型

图 $G=(V,E)$ 表示节点集合为 V，边集合为 E 的网络拓扑结构。对于网络中任意节点 $\forall v \in V$，其对应的邻居集合可以表示为 $N(v)$，以其为根的最短路径树可以表示为 $spt(v)$。

$D(v,x)$ 为在 $spt(v)$ 中节点 x 的所有子孙节点（包含节点 x）。对于 $\forall(i,j)\in E$，$w(i,j)$ 是该链路代价；对于 $\forall x,y \in V\ (x \neq y)$，$cost(x,y)$ 是节点 x 到节点 y 的最小代价。$dn(x,y)$ 是节点 x 到节点 y 的最优下一跳，$bn(x,y)$ 是节点 x 到节点 y 的备份下一跳的集合。

图 1 表示以节点 c 为根的最短路径树 $spt(c)$。节点 c 的邻居节点可以表示为 $N(c)=\{a,b\}$。在 $spt(c)$ 中，节点 b 的所有子孙节点可以表示为 $D(c,b)=\{e,f\}$，节点 a 的所有子孙节点可以表示为 $D(c,a)=\{h,d,g\}$。节点 c 到节点 g 的最短路径的代价为 $cost(c,g)=3+2+1=6$。节点 c 到节点 g 的默认下一跳为 $dn(c,g)=a$，节点 c 到节点 f 的默认下一跳为 $dn(c,f)=b$。

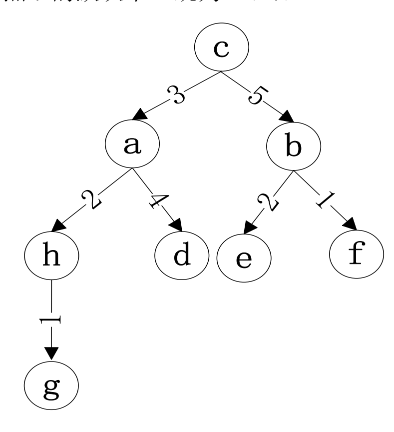


图 1　以节点 c 为根的最短路径树 $spt(c)$

Fig. 1　Shortest path tree rooted at node c

1.2　问题描述

为了提升网络可用性，提高用户体验，IETF 提出利用 LFA 规则应对网络中的故障。下面将分别介绍 LFA 中的三个规则：

a)LFC。对于任意目的地址 d，节点 c 可以将报文发送给其邻居节点 x，当且仅当满足 $cost(x,d)<cost(x,c)+cost(c,d)$。

b)NPC。对于任意目的地址 d，节点 c 可以将报文发送给其邻居节点 x，当且仅当满足 $cost(x,d)<cost(x,f)+cost(f,d)$，其中 f=$dn(c,d)$，即 f 代表节点 c 到目的地址 d 的最优下一跳。

c)DC。对于任意目的地址 d，节点 c 可以将报文发送给其邻居节点 x，当且仅当满足 $cost(x,d)<cost(c,d)$。

为了在节点 c 上实现 LFC 规则，节点 c 需要知道 $cost(c,d)$，$cost(x,c)$ 和 $cost(x,d)$ 的数值。为了在节点 c 上实现 NPC 规则，节点 c 需要知道 $cost(x,d)$，$cost(x,f)$ 和 $cost(f,d)$ 的数值。实现 DC 规则，节点 c 需要知道 $cost(c,d)$ 和 $cost(x,d)$ 的数值。节点 c 可以通过 $spt(c)$ 计算出 $cost(x,c)$、$cost(c,d)$ 和 $cost(f,d)$ 的值，但是无法获得 $cost(x,d)$ 和 $cost(x,f)$ 的数值。因此，本文需要解决的问题可以描述为：对于节点 c，如果给定 $spt(c)$，是否可以找到一个算法快速计算出其所有邻居对应的 $cost(x,d)$ 和 $cost(x,f)$，$x \in N(c)$ 的数值，其中 $d \in V, d \neq x, d \neq f$。定理 1 回答了如何解决上述的问题，并且给出了证明。

定理 1　已知计算节点 c 和以其为根的最短路径树 $spt(c)$，对于网络中的任意节点 $d(d \neq c, d \neq x)$，其中 $x \in N(c)$，$spt'(c)$ 表示将链路 (c,x) 的代价调整为 $-cost(c,x)$ 时对应的新的以节点 c 为根最短路径树。

如果 $d \in D(spt(c),x)$ 成立，则 $cost(x,d)$=$cost(c,d)$-$cost(c,x)$；

如果 $d \notin D(spt(c),x)$ 和 $d \in D(spt'(c),x)$ 同时成立，则 $cost(x,d)$=$cost'(c,d)+cost(c,x)$，$cost'(c,d)$ 表示在 $spt'(c)$ 中节点 c 和节点 d 之间的代价；

如果 $d \notin D(spt(c),x)$ 和 $d \notin D(spt'(c),x)$ 同时成立，则 $cost(x,d)$=$cost(x,c)+cost(c,d)$。

证明:

因为 $d \in D(spt(c),x)$，所以 $cost(c,d)=cost(c,x)+cost(x,d)$，即 $cost(x,d)$=$cost(c,d)$-$cost(c,x)$。

因为 $d \notin D(spt(c),x)$ 和 $d \in D(spt'(c),x)$ 同时成立，所以 $cost'(c,d)=cost'(c,x)+cost'(x,d)$，即 $cost'(x,d)=cost'(c,d)-cost'(c,x)$=$cost'(c,d)$+$cost(c,x)$。由于 $d \in D(spt'(c),x)$，所以节点 x 到节点 d 的最短路径不经过节点 c，因此 $cost'(x,d)=cost(x,d)$，即 $cost(x,d)=cost'(c,d)$+$cost(c,x)$。

因为 $d \notin D(spt(c),x)$ 和 $d \notin D(spt'(c),x)$ 同时成立，所以 $cost(x,d) \geq cost(x,c)$+$cost(c,d)$。这是因为如果 $cost(x,d)<cost(x,c)$+$cost(c,d)$ 成立，则 $cost(x,d)$-$cost(x,c)<cost(c,d)$。在 $spt'(c)$ 中，如果 $cost(x,d)$-$cost(x,c)<cost(c,d)$，则 $d \in D(spt'(c),x)$，与 $d \notin D(spt'(c),x)$ 矛盾。因此 $cost(x,d)$=$cost(x,c)+cost(c,d)$。

2　算法

2.1　算法描述

Algorithm ERPISPF

Input：$G=(V,E)$ 和 $spt(c)$.

output: $\forall v \in V$　$bn(c,v)$.

1: **for**　$x \in N(c)$　do

2:　　$weight \leftarrow cost(c,x)$

3:　　$cost(c,x) \leftarrow -cost(c,x)$

4:　　利用 i-SPF 算法构造 $spt'(c)$

5:　　根据定理 1 计算 $cost(x,d), d \in V, d \neq c, d \neq x$

6:　　$cost(c,x) \leftarrow weight$

7: endfor

8: **for**　$d \in V$　do

9:　　**for**　$x \in N(c)$　do

10:　　　**if** lfa 条件成立　**then**

11:　　　　$bn(c,d) \leftarrow bn(c,d) \cup \{x\}$

12:　　　endif

13:　　endfor

14: endfor

15：**return**　$bn(c,v), \forall v \in V$

算法 **ERPISPF** 说明了节点 c 如何实现 LFA 规则的过程。算法需要以 $G=(V,E)$ 和 $spt(c)$ 为输入条件，算法执行完毕的时候返回节点 c 到所有节点的 LFA 下一跳。访问节点 c 的邻居节点 x，调整二者之间链路的代价为 $-cost(c,x)$（算法第 2-3 行），利用增量最短路径优先算法计算节点 c 为根的最短路径树 $spt'(c)$（算法第 4 行），接着根据定理 1 计算 $cost(x,d), d \in V, d \neq c, d \neq x$（算法第 5 行），最后调整链路 (c,x) 的代价为原来的数值（算法第 6 行）。利用 LFA 条件计算节点 c 到网络中其余所有节点的备份下一跳集合（算法第 8-14 行）。算法第 10 行中的 LFA 条件表示 LFC、NPC 或者 DC，修改这里的条件就可以计算出满足不同规则的备份下一跳。

2.2　算法性能分析

定理 2　**ERPISPF 的运行复杂度为** $O(|E|lg|V|)$。

证明　假设节点 c 运行 **ERPISPF** 算法，从上述的程序可

以看出该节点需要执行 k 次最短路径优先算法，k 代表 c 的度数。如果用 $M(l)$ 代表执行算法时变化的节点的数量，$P(l)$ 代表执行算法时变化的边的数量，因此，ERPISPF 的运行复杂度为 $\sum_{l=1}^{k} M(l)lgM(l) \le lg|V|*\sum_{l=1}^{k} M(l) = O(|E|lg|V|)$ 。

定理 3　算法 ERPISPF 可以计算出所有满足 LFA 规则的下一跳集合。

证明　假设节点 c 运行该算法，从算法的第 1 行到第 7 行可知，当该部分程序执行完毕后，算法可以计算出节点 c 的所有邻居节点到其他节点的最小代价。只要知道了节点 c 的所有邻居节点到其他节点的最小代价，就可以利用 LFA 规则计算节点 c 到其他节点的备份下一跳。因此该定理成立。

3　实验及结果

由于实现 LFA 中三个规则的算法是类似的，因此本文仅说明实现 LFC 规则的算法，并且在实验中用 **ERPISPF** 表示。

3.1　实验中数据集合

1）真实拓扑

a）Abilene[27]，该拓扑包括 11 个路由器和 28 条数据链路。

b）表 2 列出了 Rocketfuel[28]项目发布的真实拓扑。

表 1　Rocketfuel 拓扑参数

Table 1　Rocketfuel parameters

AS 号码	AS 名称	结点数量	链路数量
1221	Telstra	108	153
1239	Sprint	315	972
1755	Ebone	87	162
3257	Tiscali	161	328
3967	Exodus	79	147
6461	Abovenet	128	372

2）模拟拓扑

本文利用 Brite[29]软件模拟生成网络拓扑结构，该软件的参数在表 2 中列出。

表 2　Brite 生成拓扑结构的参数设置

Table 2　Brite parameters

模型	节点数量	HS	LS
Waxman	400	1000	100
链路节点比	NodePlacement	增长方式	alpha
2-10	Random	增量式	0.15
beta	BWDist	BwMin-BwMax	模式
0.2	Constant	10.0-1024.0	路由器

3.2　评价指标

本文的评价指标主要有计算开销和故障保护率。

1)计算开销

计算开销=算法的实际运行时间/迪杰斯特拉算法的实际运行时间。

2)故障保护率

$$故障保护率=\frac{\sum_{s,d\in V} k(s,d)}{|V|*(|V|-1)}，\quad k(s,d)=\begin{cases}1, & if\ bn(s,d)\ne\Phi\\ 0, & else\end{cases}$$

3.3　计算开销

图 2 描绘了 ERPISPF、LFC、TBFH 和 DMPA 四种算法在 Abilene 和 Rocketfuel 拓扑中的计算开销结果。根据图 2 可知，在所有运行的拓扑结构中，ERPISPF 都拥有最高的执行效率，其次是 DMPA 和 TBFH，LFC 的执行效率是最差的。这是因为 ERPISPF，DMPA 和 TBFH 的计算开销并不会随网络度的变化而变化。根据定理 2 可知，ERPISPF 的计算开销小于计算一棵最短路径树的时间。DMPA 的计算开销等于计算一棵最短路径树的时间，TBFH 的计算开销接近计算两棵最短路径树的时间。

图 3 表示在模拟网络中，网络拓扑大小和计算开销之间的关系。从图 3 可知，四种算法的计算开销与网络拓扑都没有关系，ERPISPF 的计算开销仍然的四种算法中最小的。这是因为当网络拓扑大小增加的时候，算法的实际运行时间和迪杰斯特拉算法的实际运行时间都会增加，并且二者增加的比例几乎接近，所以四种算法的计算开销与网络拓扑大小几乎没有关系。

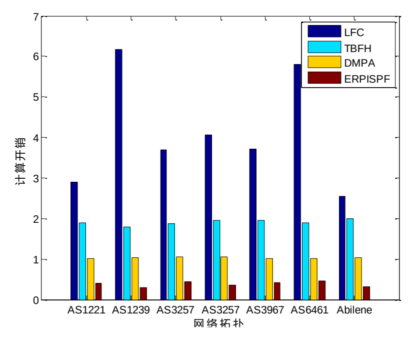


图 2　在 Abilene 和 Rocketfuel 拓扑中不同算法的计算开销

Fig. 2　Computational overhead of different algorithms on Abilene and Rocketfuel topologies

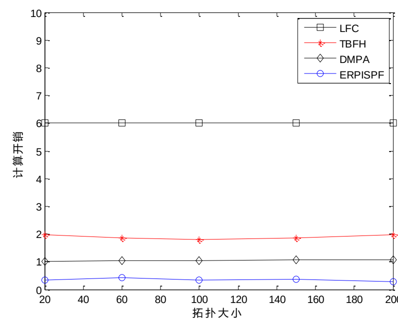


图 3 计算开销和网络拓扑大小的关系

Fig. 3　Relationship between computational overhead and topology size

图 4 表示在模拟网络中，网络节点平均度和计算开销之间的关系。从图 4 可以看出，LFC 的计算开销和节点平均度呈现线性关系，其余三种算法与节点平均度几乎没有关系。该原因和图 2 的原因是一样的。

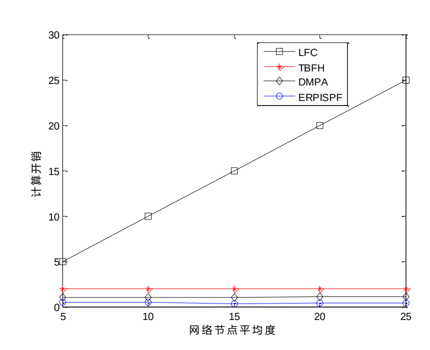


图 4　计算开销和节点平均度的关系

Fig. 4　Relationship between computational overhead and average node degree

3.4 故障保护率

图 5 描绘了 ERPISPF、LFC、TBFH 和 DMPA 四种算法在 Abilene 和 Rocketfuel 拓扑中的故障保护率结果。根据图 5 可知，在所有运行的拓扑结构中，ERPISPF 和 LFC 的故障保护率是最高的，其次是 DMPA 和 TBFH。这是因为根据定理 3 可知，ERPISPF 可以计算出所有符合 LFC 规则的下一跳。由于 DMPA 和 TBFH 实现方法的局限性，并不能完整的计算出所有符合规则的下一跳。

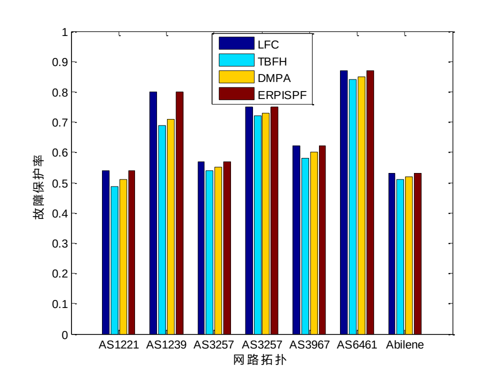


图 5　在 Abilene 和 Rocketfuel 拓扑中不同算法的故障保护率

Fig. 5　Failure protection rate of different algorithms on on Abilene and Rocketfuel topologies

图 6 表示在模拟网络中，网络拓扑大小和故障保护率之间的关系；图 7 表示在模拟网络中，网络节点平均度和故障保护率之间的关系。从两个图中可以看出，LFC 和 ERPISPF 的故障保护率始终是相同的。根据图 7 可知，故障保护率和网络节点平均度呈现出接近线性的关系。这是因为网络节点平均度增加时，每个节点的邻居节点数量随之增加，符合 LFC 规则的下一跳数量也随之增加。

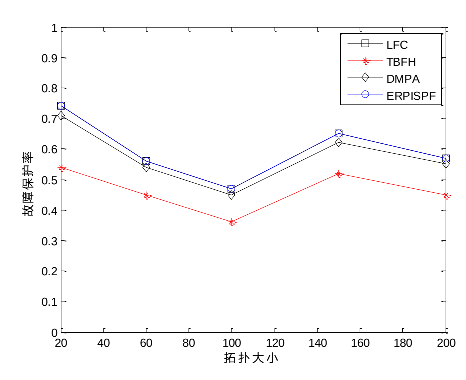


图 6　故障保护率和网络拓扑大小的关系

Fig. 6　Relationship between failure protection rate and topology size

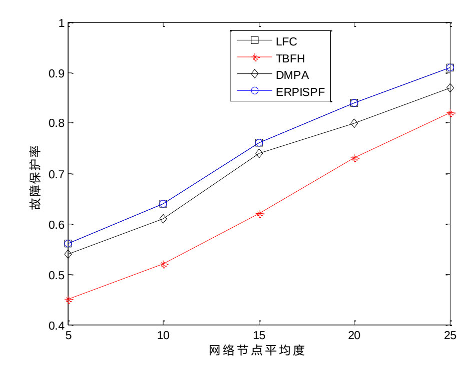


图 7　故障保护率和节点平均度之间的关系

Fig. 7　Relationship between failure protection rate and average node degree

4　结束语

为了提高 LFA 算法的计算效率，本文提出了一种基于增量最短路径优先算法的高效 LFA 实现方法。本文充分利用了增量最短路径优先的特性，巧妙地在一棵最短路径树上求解出根节点的邻居到其余所有节点的最小代价，从而快速实现 LFA。最后在 Abilene、Rocketfuel 和 Brite 生成拓扑中模拟了该算法，结果表明 ERPISPF 的计算开销小于计算一棵最短路径树的时间，并且可以计算出所有符合 LFA 规则的下一跳。

参考文献：

[1] Abishek Gopalan, Srinivasan Ramasubramanian. IP fast rerouting and disjoint multipath routing with three edge-independent spanning trees [J]. IEEE/ACM Trans on Networking, 2016, 24(3): 1336-1349.

[2] Antonakopoulos S, Bejerano Y, Koppol P. Full protection made easy: the dispath ip fast reroute scheme [J]. IEEE/ACM Trans on Networking, 2015, 23(4): 1229-1242.

[3] Xu An, Bi Jun, Zhang Baobao. Failure inference for shortening traffic detours [C]//Proc of International Symposium on Quality of Service. Piscataway, NJ: IEEE Press, 2016: 1-10.

[4] Krist P. Scalable and efficient multipath routing: complexity and algorithms [C]//Proc of IEEE International Conference on Network Protocols, Piscataway, NJ: IEEE Press, 2015: 376-385.

[5] Yang Yuan, Xu Mingwei, Li Qi. Tunneling on demand: a lightweight approach for IP fast rerouting against multi-link failures [C]//Proc of International Symposium on Quality of Service. Piscataway, NJ: IEEE Press, 2016: 1-10.

[6] Elhourani T, Gopalan A, Ramasubramanian S. IP fast rerouting for multi-link failures [C]//Proc of Conference on Computer Communications. Piscataway, NJ: IEEE Press, 2014: 2148-2156.

[7] Narváez P, Siu K Y, Tzeng H Y. New dynamic algorithms for shortest path tree computation [J]. IEEE/ACM Trans on Networking, 2000, 8(6): 734-746.

[8] Narváez P, Siu K Y, Tzeng H Y. New dynamic SPT algorithm based on a ball-and-string model [J]. IEEE/ACM Trans on Networking, 2001,9(6): 706-718.

[9] Francois P, Filsfils C, Evans J, *et al.* Achieving sub-second IGP convergence in large IP networks [J]. Computer Communication Review, 2005, 35(3): 35-44.

[10] Narvaez P. Routing reconfiguration in IP networks [D].Cambridge: Massachusetts Institute of Technology, 2000.

[11] Lakshminarayanan K, Caesar M, Rangan M, *et al.* Achieving convergence-free routing using failure-carrying packets [C]//Proc of ACM Special Interest Group on Data Communication, New York: ACM Press, 2007: 241-252.

[12] Francois P, Bonaventure O. Avoiding transient loops during the convergence of link-state routing protocols [J]. IEEE/ACM Trans on Networking, 2007, 15(6): 1280-1292.

[13] Clad F, Merindol P, Vissicchio S, *et al.* Graceful router updates for link-state protocols [C]//Proc of IEEE International Conference on Network Protocols, Piscataway, NJ: IEEE Press, 2013: 1-10.

[14] Zhang Baobao, Wu Jianping, Bi Jun. RPFP: IP fast reroute with providing complete protection and without using tunnels [C]// Proc of International Symposium on Quality of Service. Piscataway, NJ: IEEE Press,2013: 1-10.

[15] Enyedi G, Rétvári G, Szilágyi P, *et al.* IP Fast ReRoute: lightweight

not-via without additional addresses [C]// Proc of Conference on Computer Communications, Piscataway, NJ:IEEE Press,2009: 157-168.

[16] Xu Mingwei, Yang Yuan, Li Qi. Selecting shorter alternate paths for tunnel-based IP fast ReRoute in linear time [J]. Computer Networks, 2012, 56(2): 845-857.

[17] Joel Sommers, Paul Barford, Brian Eriksson. On the prevalence and characteristics of MPLS deployments in the open Internet [C]// Proc of ACM SIGCOMM Conference on Internet Measurement.New York: ACM Press, 2011: 445-462.

[18] Hao F, Kodialam M, Lakshman T V. Optimizing restoration with segment routing [C]// Proc of IEEE International Conference on Computer Communications. Piscataway, NJ: IEEE Press, 2016: 1-9.

[19] Rétvári G, Tapolcai J, Enyedi G, *et al.* IP fast ReRoute: loop free alternates revisited [C]//Proc of IEEE International Conference on Computer Communications. Piscataway,NJ: IEEE Press, 2011: 2948-2956.

[20] Yang Xiaowei, Wetherall D. Source selectable path diversity via routing deflections[C]//Proc of ACM Special Interest Group on Data Communication, New York: ACM Press, 2006: 159-170.

[21] Lee S, Yu Y, Nelakuditi S, *et al.* Proactive vs reactive approaches to failure resilient routing [C]//Proc of IEEE International Conference on Computer Communications, Piscataway, NJ: IEEE Press, 2004: 1-9.

[22] Zhong Z, Nelakuditi S, Yu Y, *et al.* Failure inferencing based fast rerouting for handling transient link and node failures [C]// Proc of IEEE Computer and Communications Societies. Piscataway, NJ: IEEE Press,2005:2859-2863.

[23] Wang Junling, Srihari Nelakuditi. IP fast reroute with failure inferencing [C]// Proc of the SIGCOMM workshop on Internet network management. New York: ACM Press,2007: 268-273.

[24] Kvalbein A, Hansen A F, Čičić T, *et al*. Fast IP network recovery using multiple routing configurations[C]//Proc of IEEE International Conference on Computer Communications Barcelona, Piscataway, NJ: IEEE Press, 2006: 1-11.

[25] Markopoulou M P, Francois P, Bonaventure O, *et al*. An efficient algorithm to enable path diversity in link state routing networks [J]. Computer Networks, 2011, 55(5): 1132-1149.

[26] Geng Haijun, Shi Xingang, Wang Zhiliang, *et al.* A hop-by-hop dynamic distributed multipath routing mechanism for link state network [J]. Computer Communications, 2018, 116: 225-239.

[27] Advanced networking for research and education, [EB/OL]. [2017-10-28] https://www. internet2. edu/products-services/advanced-networking.

[28] Spring N, Mahajan R, Wetherall D, *et al.* Measuring isp topologies with rocketfuel [J]. IEEE/ACM Trans on Networking, 2004: 2-16.

[29] Brite, [EB/OL]. [2017-10-20]. http://www. cs. bu. edu/brite/.